

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-099439

(43)Date of publication of application : 05.04.2002

(51)Int.Cl.

G06F 9/54

G06F 1/00

(21)Application number : 2000-287033

(71)Applicant : NEC CORP

(22)Date of filing : 21.09.2000

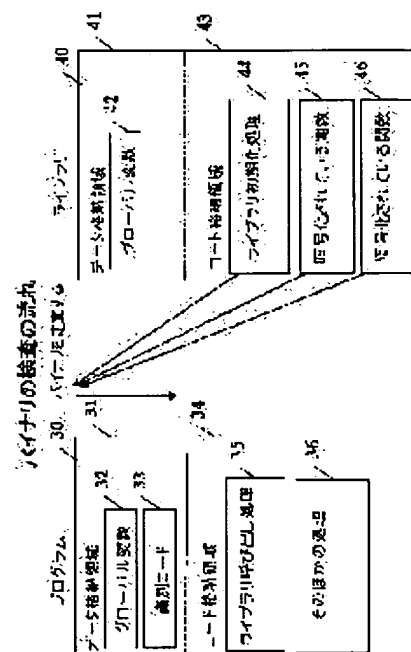
(72)Inventor : SATO TAKASHI

(54) LINK METHOD OF LIBRARY APPLICABLE TO COMPUTER SYSTEM AND RECORD MEDIUM RECORDED ITS PROGRAM

(57)Abstract:

PROBLEM TO BE SOLVED: To provide a link method of a library applicable to a computer system and a record medium recorded its program capable of calling a function by a link library only with an arbitrary executive program.

SOLUTION: In an executive program 30 to authorize a link to a library 40, a recognition code is placed, so that a key generated from the code encrypts a function that is in a high confidential among the library 40. When loading actually the library 40 from the program 30, since a control covers an initializing routine 44 in the library 40, at that time, an identification code 33 is searched with scanning a side of the program 30. Only when the code 33 in the program 30 is found, a calling of the function is made possible with expanding of functions 45 and 46 in the encrypted library.



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2002-99439

(P2002-99439A)

(43) 公開日 平成14年4月5日(2002.4.5)

(51) Int.Cl.⁷

G 0 6 F 9/54
1/00

識別記号

F I

G 0 6 F 9/06

テーマコード* (参考)

6 4 0 B 5 B 0 7 6
6 6 0 G

審査請求 有 請求項の数 8 O L (全 11 頁)

(21) 出願番号 特願2000-287033(P2000-287033)

(22) 出願日 平成12年9月21日(2000.9.21)

(71) 出願人 000004237

日本電気株式会社

東京都港区芝五丁目7番1号

(72) 発明者 佐藤 隆士

東京都港区芝五丁目7番1号 日本電気株式会社内

(74) 代理人 100088328

弁理士 金田 暢之 (外2名)

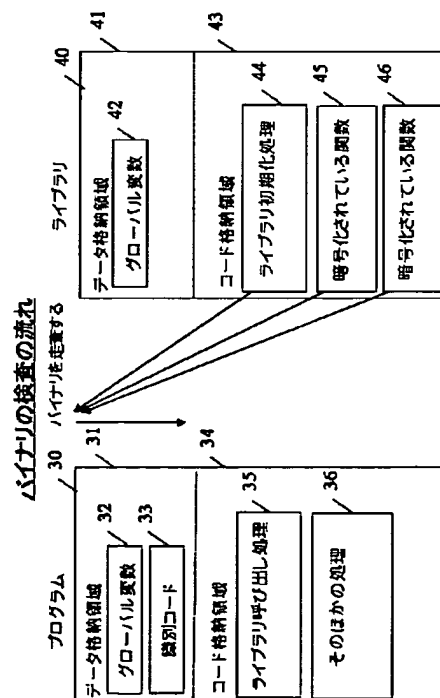
Fターム(参考) 5B076 AB04 AB15 FA20

(54) 【発明の名称】 コンピュータシステムに適用するライブラリのリンク方法及びそのプログラムを記録した記録媒体

(57) 【要約】

【課題】 リンクライブラリが任意の実行プログラムによってのみ関数を呼び出すことができるようにする、コンピュータシステムに適用するライブラリのリンク方法及びそのプログラムを記録した記録媒体を提供する。

【解決手段】 ライブラリ40にリンクを許可する実行プログラム30中に、認識コードを置いておき、その認識コードから生成した鍵で、ライブラリ中の秘密性の高い関数を暗号化しておく。実際に実行プログラム30から、ライブラリ40をロードさせた場合は、ライブラリ40中の初期化ルーチン44に制御が渡るので、そのときに、実行プログラム30側を走査して識別コード33を探す。実行プログラム30にある認識コードを見つけた場合にのみ、暗号化されているライブラリ中の関数45、46を展開して、関数の呼び出しが行えるようになる。



【特許請求の範囲】

【請求項 1】 実行対象のプログラムと予め用意されたライブラリとのリンク処理をするためのリンク機能を有するコンピュータシステムに適用するライブラリのリンク方法であって、

実行プログラム側の処理であるライブラリの動的ロードによって該ライブラリ中の初期化ルーチンと呼ぶステップと、

該初期化ルーチンにより、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、

該識別コードが正しいコードであった場合は、初期化処理を正常に完了するステップと、

前記識別コードが見つからなかった場合、または、不正な識別コードであった場合は、初期化処理を失敗とし、前記ライブラリをロードしないステップと、

実行プログラム側の処理である関数の呼び出しによって前記ライブラリ中の関数と呼ぶステップと、

該関数により、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、

該識別コードが正しいコードであった場合は、該識別コードから暗号解除の鍵を得て、暗号化された関数を解除して関数を実行するステップと、

前記識別コードが見つからなかった場合、または、不正な識別コードであった場合は、ライブラリが不正な実行プログラムから呼ばれていると判断して、処理を中止するステップと、からなるライブラリのリンク方法。

【請求項 2】 実行対象のプログラムと予め用意されたライブラリとのリンク処理をするためのリンク機能を有するコンピュータシステムに適用するライブラリのリンク方法であって、

実行プログラム側の処理であるライブラリの動的ロードによって該ライブラリ中の初期化ルーチンと呼ぶステップと、

該初期化ルーチンにより、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、

該識別コードが正しいコードであった場合は、該識別コードから暗号解除の鍵を得て、暗号化された関数を解除して初期化処理を正常に完了するステップと、

前記識別コードが見つからなかった場合、または、不正な識別コードであった場合は、初期化処理を失敗とし、前記ライブラリをロードしないステップと、

実行プログラム側の処理である関数の呼び出しによって前記ライブラリ中の関数と呼ぶステップと、

該関数により、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、

該識別コードが正しいコードであった場合は、該識別コードから暗号解除の鍵を得て、暗号化された関数を解除して関数を実行するステップと、

前記識別コードが見つからなかった場合、または、不正な識別コードであった場合は、ライブラリが不正な実行プログラムから呼ばれていると判断して、処理を中止するステップと、からなるライブラリのリンク方法。

【請求項 3】 実行対象のプログラムと予め用意されたライブラリとのリンク処理をするためのリンク機能を有するコンピュータシステムに適用するライブラリのリンク方法であって、

実行プログラム側の処理である関数の呼び出しによって前記ライブラリ中の関数と呼ぶステップと、

該関数により、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、

該識別コードが正しいコードであった場合は、該識別コードから暗号解除の鍵を得て、暗号化された関数を解除して関数を実行するステップと、

前記識別コードが見つからなかった場合、または、不正な識別コードであった場合は、ライブラリが不正な実行プログラムから呼ばれていると判断して、処理を中止するステップと、からなるライブラリのリンク方法。

【請求項 4】 前記識別コードは、前記実行プログラムのデータ格納領域またはコード格納領域またはリソース格納領域に格納されている請求項 1 または 2 に記載のライブラリのリンク方法。

【請求項 5】 実行対象のプログラムと予め用意されたライブラリとのリンク処理をするためのリンク機能を有するコンピュータシステムにより読み取り可能な記録媒体であって、

実行プログラム側の処理であるライブラリの動的ロードによって該ライブラリ中の初期化ルーチンと呼ぶステップと、

該初期化ルーチンにより、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、

該識別コードが正しいコードであった場合は、初期化処理を正常に完了するステップと、

前記識別コードが見つからなかった場合、または、不正な識別コードであった場合は、初期化処理を失敗とし、前記ライブラリをロードしないステップと、

実行プログラム側の処理である関数の呼び出しによって前記ライブラリ中の関数と呼ぶステップと、

該関数により、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、

該識別コードが正しいコードであった場合は、該識別コードから暗号解除の鍵を得て、暗号化された関数を解除して関数を実行するステップと、

前記識別コードが見つからなかった場合、または、不正な識別コードであった場合は、ライブラリが不正な実行プログラムから呼ばれていると判断して、処理を中止するステップとを、前記コンピュータシステムが実行するように設定されたプログラムを記録した記録媒体。

【請求項6】 実行対象のプログラムと予め用意されたライブラリとのリンク処理をするためのリンク機能を有するコンピュータシステムにより読み取り可能な記録媒体であって、

実行プログラム側の処理であるライブラリの動的ロードによって該ライブラリ中の初期化ルーチンを呼ぶステップと、

該初期化ルーチンにより、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、

該識別コードが正しいコードであった場合は、該識別コードから暗号解除の鍵を得て、暗号化された関数を解除して初期化処理を正常に完了するステップと、

前記識別コードが見つからなかった場合、または、不正な識別コードであった場合は、初期化処理を失敗とし、前記ライブラリをロードしないステップと、

実行プログラム側の処理である関数の呼び出しによって前記ライブラリ中の関数を呼ぶステップと、

該関数により、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、

該識別コードが正しいコードであった場合は、該識別コードから暗号解除の鍵を得て、暗号化された関数を解除して関数を実行するステップと、

前記識別コードが見つからなかった場合、または、不正な識別コードであった場合は、ライブラリが不正な実行プログラムから呼ばれていると判断して、処理を中止するステップとを、前記コンピュータシステムが実行するように設定されたプログラムを記録した記録媒体。

【請求項7】 実行対象のプログラムと予め用意されたライブラリとのリンク処理をするためのリンク機能を有するコンピュータシステムにより読み取り可能な記録媒体であって、

実行プログラム側の処理である関数の呼び出しによって前記ライブラリ中の関数を呼ぶステップと、

該関数により、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、

該識別コードが正しいコードであった場合は、該識別コードから暗号解除の鍵を得て、暗号化された関数を解除して関数を実行するステップと、

前記識別コードが見つからなかった場合、または、不正な識別コードであった場合は、ライブラリが不正な実行プログラムから呼ばれていると判断して、処理を中止するステップとを、前記コンピュータシステムが実行するように設定されたプログラムを記録した記録媒体。

【請求項8】 前記識別コードは、前記実行プログラムのデータ格納領域またはコード格納領域またはリソース格納領域に格納されている請求項4から7のいずれか1項に記載の、前記コンピュータシステムが実行するように設定されたプログラムを記録した記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明は、コンピュータシステムに適用するライブラリのリンク方法及びそのプログラムを記録した記録媒体に関する。

【0002】

【従来の技術】 ソフトウェア開発では、同じプログラムを繰り返し作成する手間を省くために、似た働きのプログラムをまとめてライブラリとして用意する。ライブラリを使うプログラムは、コンパイル時にライブラリとまとめてコンパイルを行うことで、関数宣言だけでライブラリ中のプログラムコードを使用することができる。

【0003】 ライブラリは、生成される実行プログラムの容量が増大するという点と、ライブラリの機能を使うプログラム毎にコンパイルをしなくてはならないという欠点があった。これらを解決するために用いられるのが、動的なリンクライブラリである。

【0004】 動的なリンクライブラリは、プログラムコードの実行中に動的に呼び出される。プログラムの要求に応じてシステムはライブラリをメモリ上にロードする。プログラムはそのライブラリ中の関数を呼び出して、必要がなくなったら開放する。この方式により、どのプログラムコードでもコンパイルすることなくライブラリを使用することができ、しかも必要に応じてロードするので、実行プログラムのサイズを圧迫することはない。

【0005】

【発明が解決しようとする課題】 しかし、ある特定の実行ファイルのみライブラリの使用を許したいという場合には、この動的リンクライブラリの利点が、問題になる。どんなプログラムでも関数宣言さえできればライブラリ中のプログラムを使用することができるので、秘密性の高い関数や、ある特定のプログラムでないと正常に動作しないという関数の場合でも、他のプログラムから呼び出すことが可能になるからである。

【0006】 ライブラリが実行プログラムを選ぶシステムには、これまで、リンクによってプログラムを自動リンクさせる特開平06-083593号（1994-03-25）に開示された発明や、データ処理システムに設定されたセキュリティレベルに応じて、カーネルがサーバから動的リンクするライブラリを選択する特開平08-241207号（1996-09-17）に開示された発明などがあつた。しかし前者は未解決の参照を探して解決するためのシステムであり、本発明の、特定プログラム以外とリンクをさせないという目的には用いることができない。後者はオペレーティングシステムを変更する必要がある、すでにあるシステムには対応できないという問題がある。

【0007】 また、CADプログラムに対応する暗号化されたCADライブラリの内容をアクセスした場合に、正式なライセンスに基づくアクセス要求に対してのみ復号化したCADライブラリの情報を送出するCADライ

ブラリアクセス管理装置が特開2000-76313号(2000-03-14)に開示されている。そのCADライブラリアクセス管理装置の構成を図7に示す。CADライブラリアクセス管理装置100は、ファイルシステム200内に組み込まれているが、ファイルシステム200において、リンクの許可を行う処理の大部分を行う。即ち、オペレーティングシステムなどに手を加える、もしくはそのようなオペレーティングシステムを用意する必要がある。また、図7からわかるように、ライブラリにおけるリンクの許可の有無を記載したライセンス情報60やアクセス権管理対象パス定義情報50などを別途用意する必要がある。

【0008】本発明の目的は、以上の問題を解決する、リンクライブラリが任意の実行プログラムによってのみ関数を呼び出すことができるようにする、コンピュータシステムに適用するライブラリのリンク方法及びそのプログラムを記録した記録媒体を提供することにある。

【0009】

【課題を解決するための手段】本発明のコンピュータシステムに適用するライブラリのリンク方法は、実行対象のプログラムと予め用意されたライブラリとのリンク処理をするためのリンク機能を有するコンピュータシステムに適用するライブラリのリンク方法であって、実行プログラム側の処理であるライブラリの動的ロードによってライブラリ中の初期化ルーチンと呼ぶステップと、初期化ルーチンにより、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、識別コードが正しいコードであった場合は、初期化処理を正常に完了するステップと、識別コードが見つからなかった場合、または、不正な識別コードであった場合は、初期化処理を失敗とし、ライブラリをロードしないステップと、実行プログラム側の処理である関数の呼び出しによってライブラリ中の関数と呼ぶステップと、関数により、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、識別コードが正しいコードであった場合は、識別コードから暗号解除の鍵を得て、暗号化された関数を解除して関数を実行するステップと、識別コードが見つからなかった場合、または、不正な識別コードであった場合は、ライブラリが不正な実行プログラムから呼ばれていると判断して、処理を中止するステップと、からなる。

【0010】プログラムで使うライブラリには動的リンクライブラリというものがある。動的リンクライブラリは、プログラムの実行中に必要に応じてメモリ上にロードされ、ライブラリ中の関数を呼び出すことができる仕組である。動的リンクライブラリは一般に、実行プログラムの容量を節約するためや、よく使う関数を複数の実行プログラムで共有できるといった目的で使用される。

【0011】本発明は、この動的リンクライブラリが、任意の実行プログラムからしかリンクできない機構を提

供する。これにより、第三者に使用されたくない秘密性の高い関数を、任意の実行プログラムのみから動的にリンクできるライブラリとして利用することができる。

【0012】ライブラリにリンクを許可する実行プログラム中に、認識コードを置いておき、その認識コードから生成した鍵で、ライブラリ中の秘密性の高い関数を暗号化しておく。

【0013】実際に実行プログラムから、ライブラリをロードさせた場合は、ライブラリ中の初期化ルーチン制御が渡るので、そのときに、実行プログラム側を走査して識別コードを探す。実行プログラムにある認識コードを見つけた場合にのみ、暗号化されているライブラリ中の関数を展開して、関数の呼び出しが行えるようにする。

【0014】これにより、識別コードのない第三者のプログラムでは、ライブラリの関数を使用することができなくなる。

【0015】また、実行プログラム側の処理であるライブラリの動的ロードによってライブラリ中の初期化ルーチンと呼ぶステップと、初期化ルーチンにより、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、識別コードが正しいコードであった場合は、識別コードから暗号解除の鍵を得て、暗号化された関数を解除して初期化処理を正常に完了するステップと、識別コードが見つからなかった場合、または、不正な識別コードであった場合は、初期化処理を失敗とし、ライブラリをロードしないステップと、実行プログラム側の処理である関数の呼び出しによってライブラリ中の関数と呼ぶステップと、関数により、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、識別コードが正しいコードであった場合は、識別コードから暗号解除の鍵を得て、暗号化された関数を解除して関数を実行するステップと、識別コードが見つからなかった場合、または、不正な識別コードであった場合は、ライブラリが不正な実行プログラムから呼ばれていると判断して、処理を中止するステップと、からなってもよい。

【0016】即ち、初期化ルーチンの時に識別コードから鍵を生成して、関数の暗号化を解除してもかまわない。関数が呼ばれたときにも暗号化を解除するので、二重の暗号化をかけることができる。

【0017】また、実行プログラム側の処理である関数の呼び出しによってライブラリ中の関数と呼ぶステップと、関数により、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、識別コードが正しいコードであった場合は、識別コードから暗号解除の鍵を得て、暗号化された関数を解除して関数を実行するステップと、識別コードが見つからなかった場合、または、不正な識別コードであった場合は、ライブラリが不正な実行プログラムから呼ばれていると判断し

て、処理を中止するステップと、からなってもよい。

【0018】即ち、動的リンクライブラリでなくとも静的リンクライブラリでもかまわない。静的リンクライブラリを第三者に提供しなくてはならない場合に、そのライブラリに保護すべき内容があり、特定の実行ファイル以外にリンクさせたくない場合は、本発明の方法を使うことができる。静的リンクライブラリの場合は、初期化ルーチンが存在しないので、関数が呼ばれたときの方法だけを使うことになる。

【0019】また、識別コードは、実行プログラムのデータ格納領域またはコード格納領域またはリソース格納領域に格納されていてもよい。

【0020】即ち、実行ファイルに用意する認識コードはデータ格納領域でなくともかまわない。識別コードを複数用意して、格納する場所を分散させることで、解読されにくいシステムにすることができる。

【0021】本発明のコンピュータシステムが実行するように設定されたプログラムを記録した記録媒体は、実行対象のプログラムと予め用意されたライブラリとのリンク処理をするためのリンク機能を有するコンピュータシステムにより読み取り可能な記録媒体であって、実行プログラム側の処理であるライブラリの動的ロードによってライブラリ中の初期化ルーチンを呼ぶステップと、初期化ルーチンにより、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、識別コードが正しいコードであった場合は、初期化処理を正常に完了するステップと、識別コードが見つからなかった場合、または、不正な識別コードであった場合は、初期化処理を失敗とし、ライブラリをロードしないステップと、実行プログラム側の処理である関数の呼び出しによってライブラリ中の関数を呼ぶステップと、関数により、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、識別コードが正しいコードであった場合は、識別コードから暗号解除の鍵を得て、暗号化された関数を解除して関数を実行するステップと、識別コードが見つからなかった場合、または、不正な識別コードであった場合は、ライブラリが不正な実行プログラムから呼ばれていると判断して、処理を中止するステップとを、コンピュータシステムが実行するように設定されている。

【0022】また、実行プログラム側の処理であるライブラリの動的ロードによってライブラリ中の初期化ルーチンを呼ぶステップと、初期化ルーチンにより、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、識別コードが正しいコードであった場合は、識別コードから暗号解除の鍵を得て、暗号化された関数を解除して初期化処理を正常に完了するステップと、識別コードが見つからなかった場合、または、不正な識別コードであった場合は、初期化処理を失敗とし、ライブラリをロードしないステップと、実行プ

ログラム側の処理である関数の呼び出しによって前記ライブラリ中の関数を呼ぶステップと、関数により、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、識別コードが正しいコードであった場合は、識別コードから暗号解除の鍵を得て、暗号化された関数を解除して関数を実行するステップと、識別コードが見つからなかった場合、または、不正な識別コードであった場合は、ライブラリが不正な実行プログラムから呼ばれていると判断して、処理を中止するステップとを、コンピュータシステムが実行するように設定されていてもよい。

【0023】また、実行プログラム側の処理である関数の呼び出しによってライブラリ中の関数を呼ぶステップと、関数により、実行プログラム側のバイナリイメージを順に走査して、識別コードを探すステップと、識別コードが正しいコードであった場合は、識別コードから暗号解除の鍵を得て、暗号化された関数を解除して関数を実行するステップと、識別コードが見つからなかった場合、または、不正な識別コードであった場合は、ライブラリが不正な実行プログラムから呼ばれていると判断して、処理を中止するステップとを、コンピュータシステムが実行するように設定されていてもよい。

【0024】

【発明の実施の形態】図1は、一般的な動的リンクライブラリの制御の流れを示した図である。プログラム10とライブラリ20から構成されている。実行プログラム10は、ライブラリ20中の関数をコールするプログラムである。プログラム10は、システムにライブラリを動的にロードさせる、ライブラリの動的ロード11の処理と、ライブラリ中の関数22を使用するための関数の呼び出し12の処理を含んでいる。

【0025】ライブラリ20は、動的リンクライブラリである。実行プログラムからコールされる関数22、23がある。

【0026】ライブラリの動的ロード11によって、制御はライブラリに渡り、初期化ルーチン21の処理が行われる。これが終了すると、ふたたび制御はプログラム10に戻る。関数呼び出し12によって関数22の呼び出しがおこなわれると、制御はライブラリ側になり、関数22の処理を行う。これが終了するとプログラム側に制御が戻る。

【0027】図2は、本発明を実施した実行プログラムとライブラリの例である。これらはプログラム30とライブラリ40から構成されている。

【0028】プログラム30は、データ格納領域31とコード格納領域34を含み、データ格納領域はグローバル変数32と、識別コード33を含む。コード格納領域は、ライブラリ呼び出し処理35と、そのほかの処理36を含んでいる。

【0029】ライブラリ40は、データ格納領域41と

コード格納領域 43 を含み、データ格納領域はグローバル変数 42 を含む。コード格納領域はライブラリ初期化処理 44 と暗号化された関数 45 を含む。この暗号化を解除する鍵は、識別コード 33 から生成できるようにしておく。

【0030】本発明において、図 2 で構成されたシステムは、次のように動作する。

【0031】図 1 のライブラリの動的ロード 11 によってライブラリ 20 中の初期化ルーチン 21 が呼ばれる。図 3 のように、初期化ルーチン 21 は、実行プログラム側のバイナリイメージ 30 を順に走査して、識別コード 33 を探す。これが正しいコードであった場合は、初期化処理を正常に完了する。識別コードが見つからなかったもしくは、不正な識別コードであった場合は、初期化処理は失敗し、ライブラリはロードされない。

【0032】関数 22 は図 1 の関数の呼び出し 12 によって呼ばれると、プログラムのバイナリイメージ 30 を検査して、識別コード 33 を探す。この識別コードから暗号解除の鍵を得て、暗号化された関数 45 を解除して関数 22 に戻して関数を実行する。

【0033】関数 23 も同様の処理が行われる。

【0034】識別コードが見つからない場合は、ライブラリが不正な実行プログラムから呼ばれていると判断して、処理を中止する。

【0035】本発明でも図 1 の制御の流れは変わらない。ライブラリ側で行われる処理に追加して行う処理がある。本発明の全体の動作について、図 4、図 5 のフローチャートを参照して、詳細に説明する。

【0036】図 4 は、ライブラリの初期化処理が行う動作を示すフローチャートである。

・許可された実行プログラムの場合

図 1 のライブラリの動的ロード 11 によって呼ばれたライブラリ初期化ルーチン 21 では、まず自分を呼んだプログラムのバイナリコードを先頭から検査していく（ステップ A1）。

【0037】この検査によってプログラム中に置かれた識別コードを見つけたのなら A3、そうでないのなら A4 へ分岐する（ステップ A2）。

【0038】得た識別コードが正しいのどうかを検査する（ステップ A3）。

【0039】ライブラリを呼び出したプログラムは、許可されたプログラムではなかったので、初期化を失敗させる（ステップ A4）。

【0040】ライブラリを呼び出したプログラムは、許可されたプログラムだったので、初期化を成功させる（ステップ A5）。

・識別コードを持たない実行プログラムの場合

自分を呼んだプログラムのバイナリコードを先頭から検査していく（ステップ A1）。

【0041】この検査によってプログラム中に識別コー

ドが見つからないので A4 へ分岐する（ステップ A2）。

【0042】ライブラリを呼び出したプログラムは、許可されたプログラムではなかったので、初期化を失敗させる（ステップ A4）。

【0043】次に、図 5 は、プログラム実行中にライブラリの持つ関数をコールされたときの前処理の動作を示すフローチャートである。

・許可された実行プログラムの場合

図 1 の関数の呼び出し 12 によって呼ばれたライブラリ中の関数は、まず自分を呼び出したプログラムのバイナリコードを先頭から検査していく（ステップ B1）。

【0044】この検査によってプログラム中に置かれた識別コードを見つけたので B3 へ分岐する（ステップ B2）。

【0045】識別コードは正しいことを検査したら B4 へ分岐する（ステップ B3）。

【0046】識別コードから暗号解除の鍵を生成する（ステップ B4）。

【0047】その鍵で関数の一部にかけられた暗号化を解除する（ステップ B5）。

【0048】関数を呼び出したプログラムは、許可された実行プログラムだったので、関数の実行を行う（ステップ B7）。

・識別コードを持たない実行プログラムの場合

まず自分を呼び出したプログラムのバイナリコードを先頭から検査していく（ステップ B1）。

【0049】この検査によって識別コードが見つからないので B6 へ分岐する（ステップ B2）。

【0050】関数を呼び出したプログラムは、許可された実行プログラムではなかったので、関数の実行ができない（ステップ B6）。

【0051】このように、ライブラリを呼び出すプログラム中に識別コードを置き、ライブラリ中の関数を、そのコードから生成した鍵で暗号化しておくことにより、識別コードのない実行プログラムが、ライブラリを呼び出して使用することを妨げることが可能になる。

【0052】識別コードから鍵を生成する手順は、識別コードから論理演算などを複数回行って得た値を使用するのが望ましい。

【0053】（発明の他の実施の形態）

・動的リンクライブラリでなくとも静的リンクライブラリでもかまわない。

【0054】静的リンクライブラリを第三者に提供しなくてはならない場合に、そのライブラリに保護すべき内容があり、特定の実行ファイル以外にリンクさせたくない場合は、本発明の方法を使うことができる。静的リンクライブラリの場合は、初期化ルーチンが存在しないので、関数が呼ばれたときの方法だけを使うことになる。この場合、ライブラリ提供先に識別コードを知らせるの

は意味をなさないで、実行ファイル名などの既知の情報を識別コードにする必要がある。

・実行ファイルに用意する認識コードはデータ格納領域でなくともかまわない。

【0055】図6の識別コードを分散して格納した例のように、プログラム・コードのパターンや、実行ファイルに格納された画像ファイルなどに認識コードを埋めてもかまわない。

【0056】識別コードを複数用意して、格納する場所を分散させることで、解読されにくいシステムにすることができ、この場合、識別コードの種類は鍵の種類になる。関数1つにつき1種類の鍵を割り当てると、ひとつの識別コードが知られても他の暗号化した関数に影響を及ぼさないライブラリができる。また、すべての関数で複数の鍵による暗号化を行うようにすれば、単体のセキュリティ強度の高いライブラリになる。

・初期化ルーチンの時に識別コードから鍵を生成して、関数の暗号化を解除してもかまわない。

【0057】関数が呼ばれたときにも暗号化を解除するので、二重の暗号化をかけることができる。この場合は鍵は共通で、暗号化アルゴリズムが異なるものを用意する必要がある。

【0058】

【発明の効果】以上説明したように、本発明の効果は、任意の実行プログラム以外からは、その関数をコールすることのできない動的リンクライブラリの作成を可能にすることにある。

【0059】その理由は、動的リンクライブラリが初期化ルーチンを通じたとき、更に、実行プログラムから関数をコールされたときに、実行プログラム側に埋め込まれている識別コードを走査して、動的リンクを許可したプログラムかどうかを検査することにある。

【0060】そして識別コードがライブラリの暗号化を解除する鍵になっていることにある。

【0061】また、動的リンクライブラリでなくとも静的リンクライブラリでもかまわない。

【0062】静的リンクライブラリを第三者に提供しなくてはならない場合に、そのライブラリに保護すべき内容があり、特定の実行ファイル以外にリンクさせたくない場合は、本発明の方法を使うことができる。

【0063】また、実行ファイルに用意する認識コードはデータ格納領域でなくともかまわない。識別コードを複数用意して、格納する場所を分散させることで、解読されにくいシステムにすることができ、

【0064】また、初期化ルーチンの時に識別コードから鍵を生成して、関数の暗号化を解除してもかまわない。関数が呼ばれたときにも暗号化を解除するので、二

重の暗号化をかけることができる。

【図面の簡単な説明】

【図1】動的ライブラリ使用時の一般的な制御の図である。

【図2】本発明を適用したライブラリと実行ファイルのバイナリイメージの図である。

【図3】本発明を適用したライブラリ初期化ルーチンのバイナリ検査の図である。

【図4】初期化ルーチンのバイナリ検査の処理のフローチャートである。

【図5】関数が実行される前処理のフローチャートである。

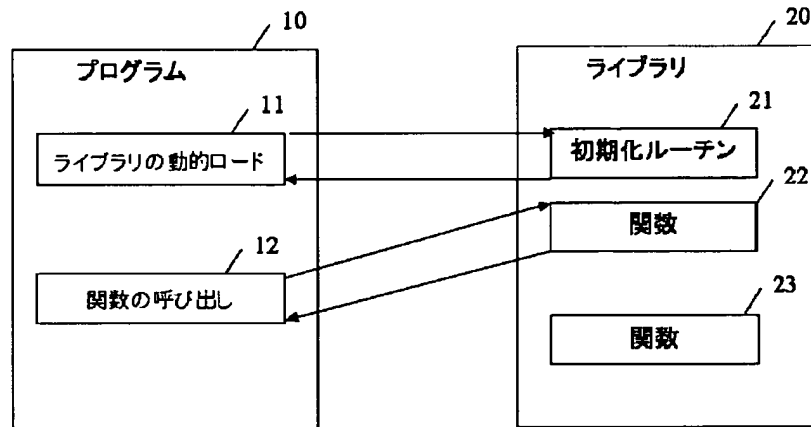
【図6】識別コードを分散して格納する場合の構成図である。

【図7】従来の技術のCADライブラリアクセス管理装置の構成を示す図である。

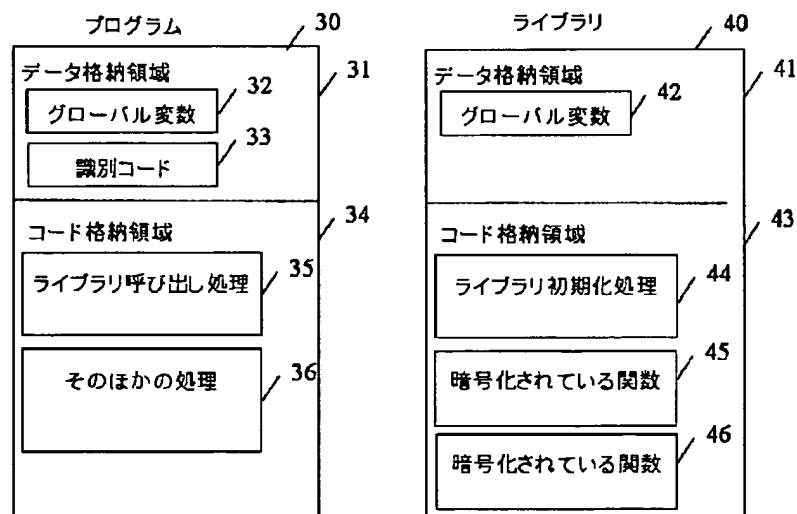
【符号の説明】

10 プログラム
11 システムにライブラリをロードさせる処理
12 ライブラリの関数を呼び出す処理
20 ライブラリ
21 初期化ルーチン
22～23 呼び出される関数
30 プログラム
31, 41 データ格納領域
32, 42 グローバル変数
33 識別コード
34, 43 コード格納領域
35 システムにライブラリをロードさせる処理
36 そのほかの処理
37 識別コードを格納したそのほかの処理
38 リソース格納領域
39 識別コードを格納した画像データ
40 ライブラリ
44 ライブラリ初期化処理
45, 46 暗号化されている関数
50 アクセス権管理対象パス定義情報
60 ライセンス情報
100 CADライブラリアクセス管理装置
102 アクセス先パス確認処理部
104 アクセス権確認処理部
106 ライブラリ復号化処理部
110 ファイルアクセス処理部
120 ファイルアクセス要求元
130 暗号化されたライブラリ
140 アクセス管理されていないファイル
200 ファイルシステム

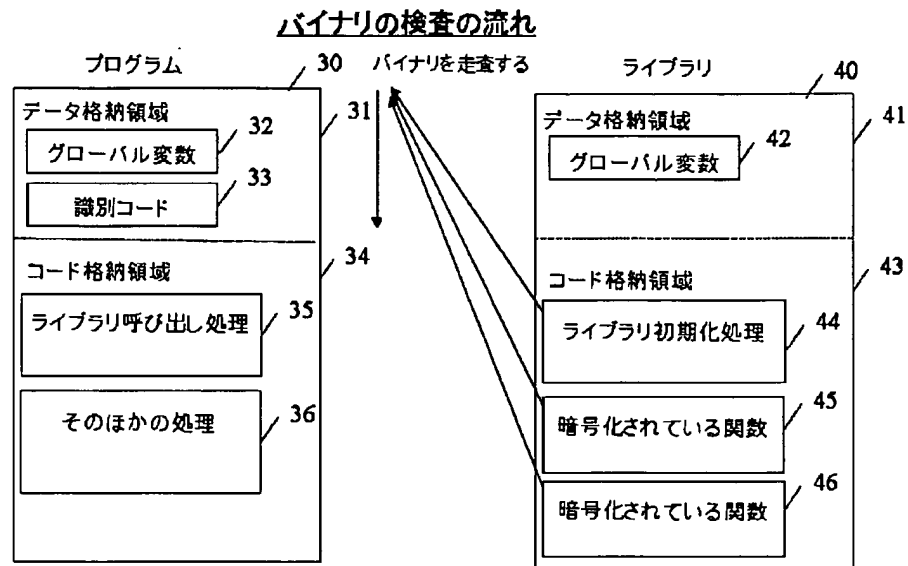
【図1】

動的ライブラリ使用時の制御の流れ

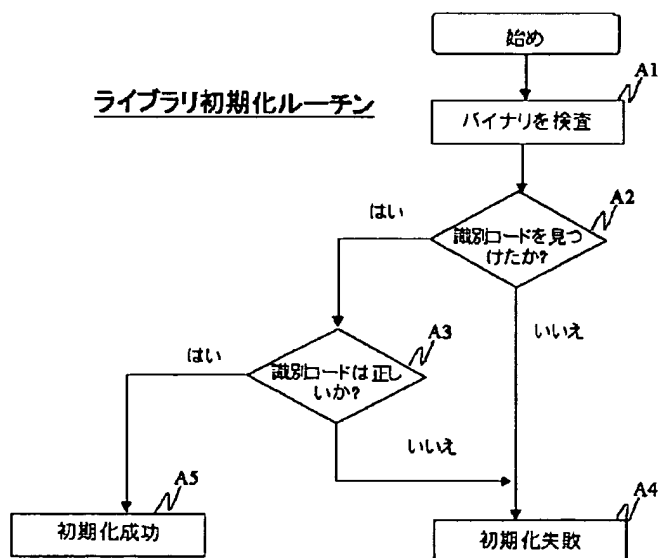
【図2】

本発明の構成

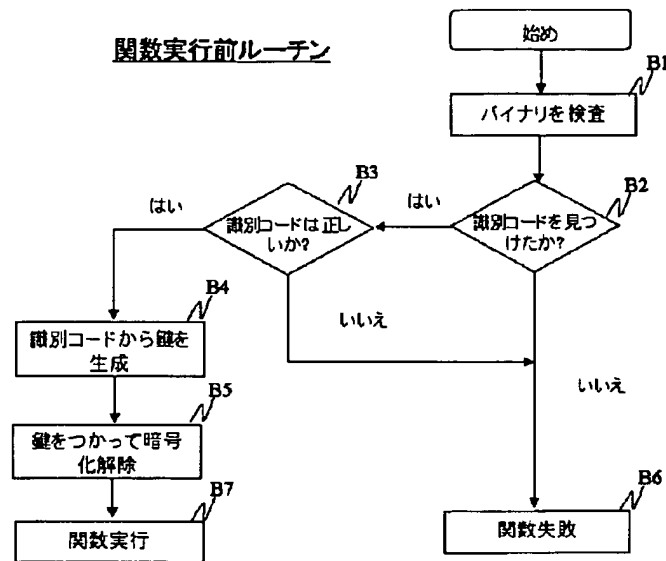
【図3】



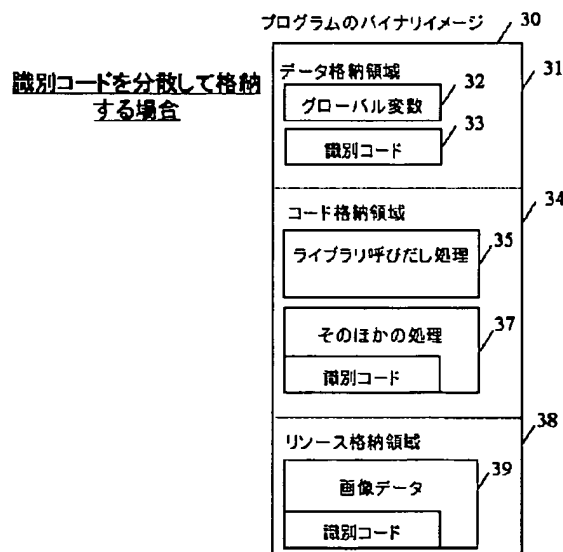
【図4】



【図5】



【図6】



【図7】

